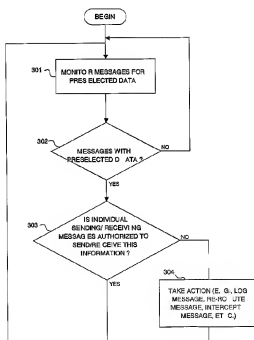(54) DETECTION DES DONNEES PRESELECTIONNEES
(54) DETECTION OF PRESELECTED DATA

(57)
A method and apparatus for detecting pre-selected data stored on a personal computing device is described. In one embodiment, contents of data storage media of a personal computing device are searched for pre-selected sensitive data. In one embodiment, if at least a portion of the pre-selected sensitive data is detected, a notification of the detection of the pre-selected data is sent to a system via a network. In another embodiment, if at least a portion of pre-selected sensitive data is detected, the access to this data is blocked.

(54) Titre : DETECTION DES DONNEES PRESELECTIONNEES
(54) Title: DETECTION OF PRESELECTED DATA
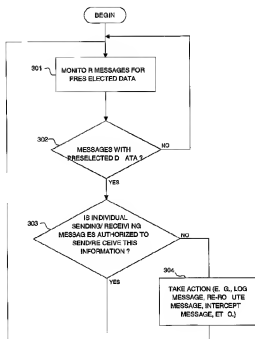
(57) Abrégé/Abstract
A method and apparatus for detecting pre-selected data stored on a personal computing device is described. In one embodiment, contents of data storage media of a personal computing device are searched for pre-selected sensitive data. In one embodiment, if at least a portion of the pre-selected sensitive data is detected, a notification of the detection of the pre-selected data is sent to a system via a network. In another embodiment, if at least a portion of pre-selected sensitive data is detected, the access to this data is blocked.

(12) **INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)**

(51) International Patent Classification[7]: **G06F 17/60**

(21) International Application Number:
PCT/US2003/030178

(22) International Filing Date:
17 September 2003 (17.09.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/247,002   18 September 2002 (18.09.2002)   US
10/431,145   7 May 2003 (07.05.2003)   US
10/607,718   27 June 2003 (27.06.2003)   US

(71) Applicant (for all designated States except US): **VONTU, INC.** [US/US]; 748 Duncan Street, San Francisco, CA 94131 (US).

(72) Inventors; and
(75) Inventors/Applicants (for US only): **ROWNEY, Kevin, T.** [US/US]; 748 Duncan Street, San Francisco, CA 94131 (US). **WOLFE, Michael, R.** [US/US]; 2511 Octavia Street, San Francisco, CA 94123 (US). **GOPALAKRISHNAN, Mythili** [US/US]; 313 30th Street, #3, San Francisco, CA 94131 (US). **FRIDMAN, Vitali** [CA/US]; 510 South Road, Belmont, CA 94002 (US). **ANSANELLI, Joseph** [—/—]; united States (US).

(74) Agent: **MALLIE, Michael, J.**; Blakely, Sokoloff, Taylor & Zafman LLP, 12400 Wilshire Boulevard, 7th Floor, Los Angeles, CA 90025 (US).

(81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PG, PH, PL, PT,

[Continued on next page]

(54) Title: DETECTION OF PRESELECTED DATA
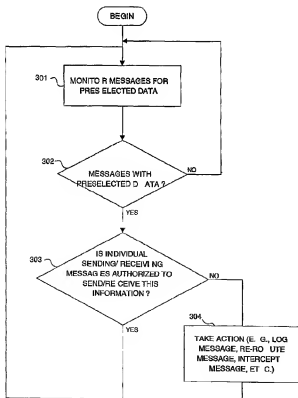
(57) Abstract: A method and apparatus for detecting pre-selected data stored on a personal computing device is described. In one embodiment, contents of data storage media of a personal computing device are searched for pre-selected sensitive data. In one embodiment, if at least a portion of the pre-selected sensitive data is detected, a notification of the detection of the pre-selected data is sent to a system via a network. In another embodiment, if at least a portion of pre-selected sensitive data is detected, the access to this data is blocked.

BEGIN

301 MONITOR MESSAGES FOR PRESELECTED DATA

302 MESSAGES WITH PRESELECTED DATA? — NO

YES

303 IS INDIVIDUAL SENDING/RECEIVING MESSAGES AUTHORIZED TO SEND/RECEIVE THIS INFORMATION? — NO

YES

304 TAKE ACTION (E. G., LOG MESSAGE, RE-ROUTE MESSAGE, INTERCEPT MESSAGE, ET C.)

WO 2004/027653 A3

RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR,
TT, TZ. UA. UG, US, UZ, VC, VN, YU. ZA, ZM. ZW.

(84) **Designated States** *(regional):* ARIPO patent (GH. GM.
KE, LS, MW, MZ, SD. SL. SZ, TZ, UG, ZM, ZW),
Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM),
European patent (AT, BE, BG, CH. CY, CZ, DE, DK, EE.
ES, FI, FR. GB, GR, HU, IE, IT, LU, MC, NL, PT, RO.
SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM.
GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*
— *before the expiration of the time limit for amending the*
  claims and to be republished in the event of receipt of
  amendments

(88) **Date of publication of the international search report:**
30 September 2004

For two-letter codes and other abbreviations, refer to the "Guid-
ance Notes on Codes and Abbreviations" appearing at the begin-
ning of each regular issue of the PCT Gazette.

SEARCHABLE INFORMATION CONTENT FOR PRESELECTED DATA

FIELD OF INVENTION

[0001]   The present invention relates to the field of processing data; more particularly, the present invention relates to detecting preselected (e.g., proprietary) data in information content.

BACKGROUND OF THE INVENTION

[0002]   Many organizations store large amounts of security-sensitive information in relational databases. This type of data is usually subjected to very thorough security measures including physical security, access control, perimeter security restrictions, and - in some cases - encryption. Since access to database data is essential to the job function of many employees in the enterprise, there are many possible points of possible theft or accidental distribution of this information. Theft of information represents a significant business risk both in terms of the value of the intellectual property as well as the legal liabilities related to regulatory compliance.

Relational Database Systems

[0003]   Relational database systems are useful for a huge range of applications. Relational structures hold data in a fashion that presents naturally intuitive ways to query the data, and has the added advantage of hiding the details of the underlying disk storage system from the user. The typical applications for database systems involve the storage and retrieval of a large number of smaller pieces of data that can be naturally formatted into a table structure. Relational databases have high utility because the types of queries that most people care about can be optimized using the well-known index structures outlined below.

[0004]   The queries requested of relational database systems use a naturally intuitive predicate logic called Structured Query Language (SQL) that allows the user to succinctly request the tabular data that she/he may be looking for. Database tables almost always come equipped with an index that makes queries based on SQL more

efficient. These indices are stored in memory using a data structure called a B-tree. The salient characteristics of B-trees most relevant to the current discussion are as follows:

> B-trees are an abstract data structure based on the binary tree;
>
> B-trees must contain some copies of the data that they index; and
>
> B-trees are most efficient using the query examples outlined below.

[0005] Here are a number of query examples:

> Exact match queries of the form A = v , where:
>
> A refers to the column or "attribute" of a given database table
>
> v refers to a specific attribute value
>
> e.g., SELECT * FROM CUSTOMERS WHERE Income = 30,000
>
> Range queries of the form v1 < A < v2, where:
>
> A refers to the column or "attribute" of a given database table
>
> e.g., SELECT * FROM CUSTOMERS WHERE 30 < Income < 40
>
> Prefix queries of the form A MATCHES s*, where:
>
> "s" refers to a specific string value
>
> "s*" is a regular expression
>
> e.g., Last_Name MATCHES "Smith*"

[0006] There are a number of references to original works in the field of database systems. The first is the seminal work on relational databases by E. F. Codd., "A Relational Model of Data for Large Shared Data Banks", Communications of the ACM, 13(6): 377-387, 1970.

[0007] The second reference is one of the first published works on the "B-Tree" data structure that is the fundamental data structure that enables efficient queries of the type outlined above. See Rudolf Bayer and Edward M. McCreight, "Organization and Maintenance of Large Ordered Indices", Record of the 1970 ACM SIGFIDET Workshop on Data Description and Access, November 15-16, 1970, Rice University, Houston, Texas, USA (Second Edition with an Appendix), pages 107-141, ACM, 1970.

Information Retrieval Systems

[0008] Information retrieval is a broad field that deals with the storage and retrieval of textual data found in documents. These systems are different from those of

2

database systems chiefly in their focus on standard documents instead of tabular data. Early examples of this system were developed as part of the SMART system at Cornell. Today, the best-known information retrieval applications are web-based search engines like Google, Inktomi, and AltaVista. The typical way to use these systems is to find a reference to a document that is part of a larger set of digital documents. The user experience for these applications usually consists of a series of queries interleaved with browsing of the results. Results of the queries are presented in order of descending relevance, and the user is able to refine the queries after further browsing. As with relational databases, the huge popularity of these systems is due to the ability of the underlying indices to deliver quick responses to the types of queries that people find most useful.

[0009]   Most of these systems are based on indices that are derived from so-called "concordances" that are built up from the collection of documents indexed. These concordances contain a data structure that lists, for each word, the location of each occurrence of that word in each of the documents. Such data structures allow quick lookups of all documents that contain a particular term. For user queries that ask for all documents that contain a collection of terms, the index is structured so that it represents a large number of vectors in Euclidean vector space of high dimension. The user's list of query terms is then also re-interpreted as a vector in this space. The query is run by finding which vectors in the document space are nearest to the query vector. This last approach has a variety of different optimizations applied to it for accuracy and speed, and is called the "cosine metric".

[0010]   As mentioned above, the typical user interaction with these sorts of systems is an iterative cycle of querying, browsing, refining, and back to querying again. Query results are usually large numbers of documents that are ranked in order of relevance, and the false positive rate can be very high. Here are some classic examples of queries.

          Boolean queries like:

          a) all documents that contain the terms "database" and "indices"

          b) all documents that contain "database" or "indices" but not "Sybase"

          Link-based queries like:

          a) all documents that are linked to by documents that contain the term "dog"

b) the most "popular" (i.e. linked to) document that contains the word "dog"

[0011]   One of the first significant implementation projects of information retrieval systems is the SMART system at Cornell. This system contains many of the essential components of information retrieval systems still in use today: C. Buckley, "Implementation of the SMART Information Retrieval System", Technical Report TR85-686, Cornell University, 1985

[0012]   The WAIS project was an early application of the massively parallel super-computer produced by Thinking Machines Inc. This is one of the first fielded information retrieval systems made available over the Internet. This primary reference source for this work is by Brewster Kahle and Art Medlar: "An Information System for Corporate Users: Wide Area Information Servers." Technical Report TMC-199, Thinking Machines, Inc., April 1991, version 3.19.

[0013]   Among the many contemporary commercial vendors of Internet search services is Google. Google's real break-through in search accuracy is its ability to harvest data from both the text of the documents that are indexed as well as the hyper-link structure. See Sergey Brin, Lawrence Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", http://dbpubs.stanford.edu:8090/pub/1998-8

File Shingling Systems

[0014]   The growth of the Internet and affordable means of copying and distributing digital documents spurred research interest in technologies that can help detect illegal or inappropriate copies of documents. The primary application for this work was to detect the violation of copyright law, and to detect plagiarism. There is also significant interest in this problem as it relates to spam-email (AKA unsolicited commercial email) detection and automatic elimination. The technical term applied to describe most of these techniques is "file shingling" in which adjacent sequences of document fragments are reduced to "shingles" by hash codes, and then stored in a lookup table in the same sequence as they are found in the document.

[0015]   File shingling provides a very quick way to look for similarity between two documents. In order to provide protection to a specific document (e.g., a text file) the document is shingled by hashing the document sentence-by-sentence and storing

these hashed sentences in a table for quick lookup. In order to test a new document to see if it contains fragments of copyrighted content, the same hash function is applied to each fragment of the test message to see if the fragments appear in a similar order as they do in the copyrighted content. The technique is quick because the time required to lookup an individual fragment can be very fast.

[0016]   The typical user interaction with a file shingling system is passive instead of active. File shingling systems are usually set up to process documents automatically and deliver the query results to a user asynchronously. A typical file shingling application might be spam prevention where a set of messages is used to create an index of restricted content that an organization does not want delivered to its email systems. In this scenario, the "query" is just the automatic processing of email messages and appropriate automatic routing.

[0017]   With respect to document equivalency queries, for each test document t, find all documents d in our collection of indexed documents that have the same contents as t. For the case of spam detection, the set d could be all of the known active spam messages, and the document t could be an incoming email message.

[0018]   With respect to cut-and-paste detection queries, for each test document t, find all documents d in our collection of indexed documents in which some fragment of d occurs in t. For the case of plagiarism detection, the set d could be all of the previously submitted essays for a particular class, and the document t could be a new paper written by a student who is suspected of plagiarism.

[0019]   The main published research projects in file shingling are called KOALA, COPS, and SCAM. They all use variants on the basic file shingling approach described above with variants that optimize performance and accuracy. For information on KOALA, see N. Heintze, "Scalable Document Fingerprinting", Proceedings of Second USENIX Workshop on Electronic Commerce, November 1996. http://www-2.cs. cmu.edu/afs/cs/user/nch/www/koala/main.html. For information on COPS, see S. Brin, J. Davis, and H. Garcia-Molina, "Copy Detection Mechanisms for Digital Documents", Proceedings of the ACM SIGMOD Annual Conference, May 1995. For information on SCAM, see N. Shivakumar and H. Garcia-Molina, "SCAM: A Copy Detection Mechanism for Digital Documents", Proceedings of 2nd International Conference in Theory and Practice of Digital Libraries (DL'95), June 1995, http://www-db.

stanford.edu/~shiva/SCAM/scamInfo.html, and also see (by N. Shivakumar and H. Garcia-Molina), "Building a Scalable and Accurate Copy Detection Mechanism", Proceedings of 1st ACM Conference on Digital Libraries (DL'96) March 1996, http://www-db. stanford. edu/pub/papers/performance.ps.

Internet Content Filtering Systems

[0020]   A variety of commercial applications, referred to as content filtering systems, implement protection measures. There are two major types of applications in this category: web site restriction/monitoring software, and email content control. In both cases, the main algorithm currently in use is pattern matching against a set of regular expressions for a set collection of text fragments that would indicate data misuse. An example might be to restrict all browsing at URLs that contain the text fragment "XXX". An example for the email content control category is stopping and blocking all email that contains the words "proprietary" and "confidential" but not the words "joke" or "kidding".

SUMMARY OF THE INVENTION

A method and apparatus for detecting pre-selected data stored on a personal computing device is described. In one embodiment, the method comprises monitoring messages electronically transmitted over a network for embedded preselected data and performing content searches on the messages to detect the presence of the embedded preselected data using an abstract data structure derived from the preselected data.

BRIEF DESCRIPTION OF THE DRAWINGS

[0021]   The present invention will be understood more fully from the detailed description given below and from the accompanying drawings of various embodiments of the invention, which, however, should not be taken to limit the invention to the specific embodiments, but are for explanation and understanding only.

[0022]   **Figure 1** illustrates one embodiment of a workflow.

[0023]   **Figures 2A** and **2B** illustrate exemplary modes of operation.

[0024]   **Figure 3** is a flow diagram of one embodiment of a process for protecting database data.

[0025]   **Figure 4** is a flow diagram of one embodiment of a process for indexing database data.

[0026]   **Figure 5** is a flow diagram of one embodiment of a process for searching information content for preselected data.

[0027]   **Figure 6** is a flow diagram of one embodiment of a process for finding a match for a subset of content fragments in an abstract data structure derived from preselected data.

[0028]   **Figures 7A – 7C** are flow diagrams of alternate embodiments of a process for searching an incoming message using a hash table index of preselected data.

[0029]   **Figure 8** is a block diagram of an exemplary computer system that may perform one or more of the operations described herein.

[0030]   **Figure 9** is a block diagram of one embodiment of a system for client-based protection of pre-selected sensitive data.

[0031]   **Figure 10** is a flow diagram of one embodiment of a process for client-based protection of pre-selected sensitive data.

DETAILED DESCRIPTION OF THE PRESENT INVENTION

[0032]   A system and methodology is described herein to track and monitor the use of sensitive information anywhere on a personal computing device. In one embodiment, this monitoring is implemented by performing content searches of data storage media of a personal computing device such as a desktop computer or a portable computer. In another embodiment, the monitoring is implemented by performing content searches on messages as they are transmitted from or received by the personal computing device. In yet another embodiment, the monitoring is implemented by performing content searches before, during, and after the use of potentially sensitive information inside any application running on the personal computing device. In one embodiment, the system described herein is able to detect this information in a secure and scalable fashion that is capable of handling large amounts of the database data. Database data may comprise any form of tabular-formatted data stored in a variety of systems including, but not limited to, relational databases, spreadsheets, flat files, etc.

7

CA 02499508 2005-03-18

[0033]   In the following description, numerous details are set forth to provide a more thorough explanation of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form, rather than in detail, in order to avoid obscuring the present invention.

[0034]   Some portions of the detailed descriptions which follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0035]   It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0036]   The present invention also relates to apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only

memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

[0037]   The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

[0038]   A machine-readable medium includes any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computer). For example, a machine-readable medium includes read only memory ("ROM"); random access memory ("RAM"); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.); etc.

Components of an Exemplary Embodiment

[0039]   In one embodiment, the system to perform the detection scheme described herein consists of two main components: a Policy Management System (PMS) and a Message Monitoring System (MMS). The PMS is responsible for accepting user input that determines information security policies for the use and transmission of data (e.g., database data) that is contained inside messages sent over the network or is stored in data storage media of the personal computing devices such as portable computers, desktop computers, Personal Digital Assistants, cell-phones, etc. This data is, thus, preselected. The term "data storage media of a personal computing device" as used herein refers to any storage within the personal computing device or accessible to the personal computing device that may store, temporarily or permanently, data for the personal computing device.

[0040]   The MMS is responsible for performing content searches on messages sent over the network, data processed by personal computing devices, or data stored on data storage media of personal computing devices, and is responsible for implementing the policy identified to the PMS by the user. In one embodiment, both of these systems are coupled to a computer network that communicates any of the standard protocols for the exchange of information.

[0041]   In the normal course of operation in this embodiment, a user may decide to implement a given policy that restricts the use or transmission of database data by certain individuals and then manually enters this policy into the PMS using a graphical-user-interface and one or more user input devices (e.g., a mouse, a keyboard, etc.). The user interface receives the input and may be running on a computer system with the PMS or on a separate machine. An example policy might be to stop a given group of individuals in customer service from saving a data file containing pre-selected data to a removable media device attached to a personal computing device. In one embodiment, the policy includes the nature of protection desired (e.g., restrict only a subset of employees), the type of data that requires protection (e.g., database data), and the network location (e.g., database table name, IP address of server, server or file name) of the database data that requires protection. Again, all of this information may be specified using a standard graphical user interface that prompts the user to enter the specific information in the correct fields.

[0042]   At regular intervals, which in one embodiment are adjustable by the user but defaulting to once per specified interval (e.g., day), the PMS queries the database and extracts copies of the database data that is to be protected and derives from that data an abstract data structure (hereafter called the "index") that is described in more detail below.

[0043]   The PMS then sends this index, along with the particulars on the policy that is to be implemented, to the MMS so that it can begin to enforce that policy. The MMS receives the index from the PMS together with the details on the policy to be enforced. The MMS uses the index and the policy information to enforce the policy specified by the user. In one embodiment, the MMS uses this index to search each of the outgoing messages (e.g., email messages, web mail messages, etc.) for the database data that is to be protected, as will be discussed in greater detail below. In another

embodiment, the MMS uses this index to search contents of data storage media of a personal computing device and/or the content of interactions between the user and the personal computing device for the database data that is to be protected, as will be discussed in more detail below.

[0044]   A summary of an exemplary workflow can be found in Figure 1, where the highest-value information is identified, policies are authored, and surveillance and enforcement are performed, leading to actionable business intelligence.

Network-Based Modes of Operation

[0045]   In one embodiment, the Message Monitoring System can be configured in one of two ways: "surveillance mode", and "enforcement mode". Figure 2 illustrates two network configurations. In surveillance mode, the MMS is placed somewhere on the network where it can watch traffic and report on violations of policy, but it is specifically not configured to block messages as they leave. This is shown in Figure 2A where the PMS has access to information. The PMS is coupled to the Internet via a switch, a tap and a firewall. The MMS monitors the network messages using the tap. In "enforcement mode", the MMS is able to watch traffic and report on violations, but it can also intercept and re-route messages so that their ultimate destination is changed. This is shown in Figure 2A where the PMS has access to information and is coupled to the Internet via a switch and a firewall. In this embodiment, the MMS monitors traffic using a series of servers and re-routes traffic to, for example, certain servers, if the MMS determines messages are likely to contain preselected information. The MMS may use different servers for each of the various layer protocols.

[0046]   Message re-routing is not mandatory. Alternatively, the MMS can be configured to just intercept and stop the outgoing message. An example policy in "enforcement mode" would be to route all messages that violate a policy to the manager of the person that violates the policy so that appropriate disciplinary action can take place.

[0047]   In both modes of operation, it is possible to install multiple MMSs, each with its own copy of the indices required to detect content. This parallel processing

configuration helps with problems of scale and with protecting multiple possible points of egress of information.

[0048]   In both configurations, the MMS is actively parsing messages that are transported using various application layer protocols (e.g., SMTP, HTTP, FTP, AIM, ICQ, SOAP, etc.).

[0049]   In one embodiment, the two subsystems (PMS and MMS) run on one Local Area Network (LAN). However, the PMS and MMS may be incorporated into the same physical or logical system. This consolidated configuration is more appropriate for reasons of control cost of goods required to produce the system.

[0050]   In yet another alternative embodiment, the PMS and MMS may not necessarily reside on the same LAN. The PMS may reside on the same LAN as the database information, but the MMS may reside on a different LAN that is separated from the LAN on which PMS resides. In this configuration, the two distinct LANs may ultimately be coupled together via the Internet but separated by firewalls, routers, and/or other network devices. This is an advantageous configuration for the case where a company wants to restrict another company that needs their database data (such as a law firm or marketing agency) from violating the first company's database data policy.

[0051]   Figure 3 is a flow diagram of one embodiment of a process for protecting database data. The process is performed by processing logic that may comprise hardware (circuitry, dedicated logic, etc.), software (such as is run on a general purpose computer system or a dedicated machine), or a combination of both.

[0052]   Referring to Figure 3, processing logic monitors messages for pre-selected data (processing block 301). Next, processing logic determines whether a message has pre-selected data (processing block 302). If not, processing transitions to processing block 301. If so, processing logic determines if the individual sending/receiving message is authorized to send/receive the information in the message (processing block 303). If so, the process ends and processing transitions to processing block 301. If not, processing logic takes one or more actions such as intercepting the message, re-routing the message, logging the message, etc. (processing block 304) and processing transitions to processing block 301.

CA 02499508 2005-03-18

Client-Based Mode of Operation

[0053]   The client-based mode of operation is directed to monitoring actions taken by a user of a personal computing device to detect user operations that may involve a potential misuse of data. These user operations may include, for example, saving or accessing restricted database data on any storage device on the computing system, using restricted database data in an application, printing restricted database data, using restricted database data in any network communication protocol, etc. In one embodiment, the monitoring of user actions is performed by parsing and searching the content that is either accessed or saved onto the local storage system of the personal computing device, or transported using various application layer protocols (e.g., SMTP, HTTP, FTP, AIM, ICQ, SOAP, etc.) In another embodiment, the monitoring of user actions is performed by intercepting and interpreting the data exchanged between the user and the personal computing device.

[0054]   Figure 9 is a block diagram of one embodiment of a system for client-based protection of pre-selected sensitive data.

[0055]   Referring to Figure 9, a server 902 communicates with client computers (referred to as clients) 910 via a network 906. The network 906 may be a private network (e.g., a local area network (LAN)) or a public network (e.g., a wide area network (WAN)). The clients 910 are computers belonging to different employees within an organization. Each client 910 may be, for example, a desktop computer, a portable computer (e.g., a laptop), or any other computer that may operate with intermittent network connectivity. A content monitoring system (also referred to herein as message monitoring system or MMS) 912 resides on each client 912 and is responsible for searching contents of data storage media of this client for pre-selected sensitive data and for intercepting and interpreting content exchanged between the user and the client 912. The data storage media may include, for example, a main memory, a static memory, a mass storage memory (e.g., a hard disk), or any other storage device that may store, temporarily or permanently, files or other documents for the client computer. In one embodiment, the MMS 912 monitors specific data operations such as file-reads, file-writes, file-updates, and read and writes to removable media devices (e.g., floppy drives, universal serial bus (USB) devices, compact disk recordable (CDR) drives, etc.). The

13

CA 02499508 2005-03-18

operation of the MMS 912 facilitates the prevention of sensitive data loss via removable and mobile devices. For example, the operation of the MMS 912 may prevent the escape of sensitive data that occurs if the user copies the sensitive data stored on the client 910 to a floppy disk, moves a file with the sensitive data to a USB-based removable memory device, prints or emails the sensitive data from the laptop or desktop computer, uses the sensitive data in an unauthorized application, etc.

[0056]   The server 902 is responsible for configuring the detection scheme described herein within the organization. The server 902 contains a PMS 904 and a message collector 914. The PMS 904 maintains a set of security policies controlling the use of sensitive data. The set of security policies may identify employees whose computers need to be monitored for a potential misuse of sensitive data, specify the sensitive data for which searches are to be performed, and define the scope of the searches (e.g., specific storage medium, data operations, etc.). Based on this information, the PMS 904 instructs each MMS 912 as to whether a corresponding client 910 is to be searched and sends the index that is to be used for searching. The index is derived from the specific sensitive data that is pre-selected for one or more clients 912 based on the security policies. The message collector 914 is responsible for collecting messages received from the MMSes 912 that notify of data misuses by the users of the clients 910.

[0057]   In one embodiment, each MMS 912 can operate in a stand-alone fashion when it cannot maintain network contact with the server 902 (e.g., if a laptop 910 is taken home for the weekend, moved to another network, stolen, etc.). For example, if the user disconnects the laptop 910 from the network 906, the MMS 912 running on the laptop 910 may perform periodic content searches of the data storage media of the laptop 910 while the user works on the laptop at home. Specifically, the MMS 912 may search the local file system of the laptop 910, an email message archive, etc. In addition, the MMS 912 may monitor specific data operations (e.g., file-reads, file-writes, file-updates, and reads and writes to removable media devices such as floppy drives) if instructed by the PMS 904. In one embodiment, when the MMS 912 detects the pre-selected data on any data storage medium of the client 910, it creates a message containing a notification of the detection of the pre-selected data, and places this message into a transmission queue. Subsequently, when the network connectivity is re-established, messages from the transmission queue are sent to the message collector 914. In one embodiment, the

14

policies maintained by the PMS 904 require that MMS 912 prevent access to the pre-selected data once the pre-selected data is detected.

[0058]   Figure 10 is a flow diagram of one embodiment of a process for personal computing device-based protection of pre-selected sensitive data. The process is performed by processing logic that may comprise hardware (circuitry, dedicated logic, etc.), software (such as is run on a general purpose computer system or a dedicated machine), or a combination of both. Processing logic resides on a personal computing device such as a client 910.

[0059]   Referring to Figure 10, processing logic receives instructions defining the scope of content searches that are to be performed on the personal computing device (processing block 1002). In one embodiment, the instructions identify data storage media that are required to be searched and the periodicity of the search. In one embodiment, the instructions also identify the data operations that are to be monitored for the presence of pre-selected sensitive data.

[0060]   Next, processing logic receives an abstract data structure or index derived from the pre-selected sensitive data (processing block 1004). Some embodiments of the abstract data structure are discussed in greater detail below.

[0061]   At processing block 1006, processing logic searches the contents of the data storage media of the personal computing device for the pre-selected sensitive data using the abstract data structure. The scope of the content search is defined by the instructions received from the server. The searching may apply to contents of data storage media of this personal computing device data and/or content exchanged between the user and the personal computing device. In one embodiment, the content search is performed periodically, at predefined time intervals. Some embodiments of the searching technique used by processing logic are discussed in more detail below. The term "data storage media of a personal computing device" refers to any form of data storage accessible to the personal computing device including, for example, magnetic disks, volatile random access memory, removable media, tape backup systems, remote network addressable storage, etc. In one embodiment, processing logic searches volatile memory devices to detect use of pre-selected data by applications running on the personal computing device. Once the use is detected, processing logic identifies the application using the pre-selected data.

15

[0062]   If processing logic detects the presence of the pre-selected data (or its portion) (processing box 1008), it then determines whether policies maintained by the PMS require blocking of access to the pre-selected data (processing box 1009). In one embodiment, the access to the detected data is blocked for the application attempting to access this data.

[0063]   If blocking is required, processing logic blocks the access to the pre-selected data (processing block 1010) and further determines whether the personal computing device can maintain network contact with the server or any other designated device (processing box 1011). If this determination is positive, processing logic sends a message containing a notification of the detection to the server (processing block 1012). The notification may identify the personal computing device and the detected data. In one embodiment, the notification identifies the application that was using the pre-selected data when running on the personal computing device.

[0064]   If there is no connection between the personal computing device and the server, processing logic places this message into a queue for future transmission to the server when the network connectivity is re-established (processing block 1014).

[0065]   As discussed above, the personal computing device-based monitoring allows for surveillance of content stored on and processed by a personal computing device. As opposed to existing desktop-based firewalls that perform filtering based on a protocol, the content searching described herein addresses the problem of specifically searching for traces of pre-selected database data inside a file-system, memory bank, or data in the process of being accessed by an application.

[0066]   With respect to the access control technique that prohibits unauthorized access by prompting the requestor for credentials (e.g., a password), the client-based protection of sensitive data that is described herein monitors content stored in the personal computing device after this content has been downloaded or otherwise accessed through an access control system.

[0067]   Desktop-based encryption/decryption packages systems, which typically rely on server-based mechanisms to encrypt the data and desktop-based mechanisms to decrypt it for viewing, help prevent the misuse of data by restricting the access to the cryptographic keys that decrypt the data. The client-based protection of sensitive data

16

CA 02499508 2005-03-18

that is described herein can be used to protect the data that is left "in the clear" outside of a cryptographic envelope and is, therefore, vulnerable to theft by third parties.

[0068]   As to the anti-viral solutions that are typically used to detect the presence of hostile hidden code inside attachments, the client-based protection of sensitive data that is described herein is directed to detecting the presence of pre-selected database data, and not the presence of the hidden code.

[0069]   With respect to driver-filters, which are forms of software written to drive the operation of hardware using a content filter that monitors all content sent to that personal computing device, they lack the ability to perform searches of data storage media of the personal computing device for pre-selected database data.

Security Requirements for One or More System Embodiments

[0070]   Since the embodiments of the detection system are used to enforce information security policies, the security properties of this system are paramount. In one embodiment, the chief objective of this system is to enforce security policies that are pertinent to database data. This implies that the system must be very secure in the manner in which it handles database data. If, in the process of protecting the database data, the system opens up new avenues to steal database data, then its ultimate purpose is defeated.

[0071]   In one embodiment, the MMS is deployed in such a way as to monitor and/or block the largest number of messages flowing through the network. This means installing the MMS at various points in the network where traffic is concentrated (e.g., routers, mail systems, firewalls, desktop computers, email archive systems, etc.) This means that the MMS may be installed either behind or in front-of one of these points of concentration on the network. Such placement of the system affords it an exceptional view of message and increases its utility for the organization using the system. Unfortunately, such placement also makes the MMS highly vulnerable to network-based attacks (commonly called "hacking") in which a third party uses unauthorized network access to violate the security perimeter surrounding the network to steal the data contained inside the network. Such placement makes MMS vulnerable to "hacking" attacks by the same employees who are being monitored by the MMS.

[0072]   In another embodiment, the MMS is deployed locally on a personal computing device and is responsible for performing surveillance on the use of local storage media, on the use of classified data by applications running on the personal computing device, and on network communications to and from the device. Such placement of the system affords it an exceptional view of the information accessed and used by the person operating the computing device and increases its utility for the organization using the system. However, such placement makes the MMS vulnerable to "hacking" attacks by the same employees who are being monitored by the MMS.

[0073]   The PMS's security concerns are also high in that its software directly queries the information sources in order to build the index that the MMS utilizes.

[0074]   Hence, the placement of the MMS on the network, in one embodiment, or on a personal computing device, in another embodiment, makes it exposed to attacks. These attacks can come, in one embodiment, from inside the Local Area Network (LAN) or from outside the LAN via the WAN and/or Internet link that the organization maintains. In another embodiment, the attacks can come from users of a personal computing device. The specific security concern here is that the MMS may contain valuable database data from the relational database that it is trying to protect. The concern is that hackers or users of personal computing devices may try to steal the data from the MMS instead of trying to steal it from the more-thoroughly guarded computer on which the relational database actually runs.

[0075]   A second and related security concern for the application arises in the case when the MMS is deployed at a different LAN from that in which the PMS is deployed. As mentioned above, this may be an important configuration to help implement security policy across two organizations that share database data. Here again, the information stored in the MMS is subjected to information security threats.

[0076]   Various embodiments treat these security threats directly. One aspect of novelty of these embodiments described herein is that the PMS/MMS pair that exchanges indices that contain no copies of the data that it is seeking to protect. As covered above, the PMS sends abstract data structures derived from the database data to the MMS so that it can enforce policy. One possible approach to achieve this protection is to simply copy the database into the MMS, or (equivalently from a security perspective) allow the MMS to directly query the database in order to check that the content is consistent with policy.

18

CA 02499508 2005-03-18

The problem with this approach is that it introduces significant security vulnerabilities where there were none before. In this insecure approach, the cure is worse than the disease.

[0077]   In one embodiment, the PMS creates an index from the database that contains no copies of the database data, or contains only encrypted or hashed copies of database data. Such an index may be created using a tuple-storage mechanism that provides a data structure for storing multiple tuples associated with fragments of the database data. Examples of the tuple-storage mechanism include a hash table, a vector, an array, a tree, a list, or a table in a relational database management system. In the process described below, the data stored in the indices only retains the relative placement of the elements in the database in relation to other elements. For example, in the case of a hash table, the index may store, for each fragment of the database data (e.g., a data fragment inside a database cell), the fragment's hash code together with its row number, column number and type of the column.

[0078]   Other embodiments of this same solution utilize indices that contain fragments of the intellectual property that is under protection, thus reducing the value of the solution by exposing that information to security threats. In one embodiment, the techniques described herein specifically avoid storing any representation of the data itself so that, in the case of a hacker breaking into the host that runs the MMS; the data that is exposed to theft is inconsequential.

[0079]   An alternate embodiment to that described by the process given below could be done to enhance performance. In this alternate embodiment, copies of only a small amount of frequently used strings and numbers from the database that represent a large proportion of the data in the system is still stored directly in the index along with the rest of the information on relative placement of data in the database table(s). This is done by storing copies of these common strings themselves, instead of hash codes. In this alternate approach, the system stores (for these common terms) the row numbers, column numbers, and type of the database data, but now instead of storing a hash code it stores the string itself. For the rest of the cells of the database that are not quite so common, only the row numbers, column numbers, and type of the database data are stored while specifically not storing copies of these strings. This approach uses the fact that the statistical distribution of string and numeric data in databases is often skewed so

that the most common terms account for a very large percentage of the overall volume of data stored. Storing these common terms in a separate index helps index query efficiency since the small number of common terms accounts for a large proportion of the queries, and these queries can be run using standard quick techniques from the literature (e.g., hash table lookups, bitmaps, etc.). The reason that this is not a security vulnerability is that this small number of terms that account for a disproportionate share of volume of database data are the least valuable pieces of data. The terms "John" and "Smith" are very common inside databases that contain names, but the theft of these terms is relatively worthless. In this embodiment, the system is still carefully avoiding storing copies of data of less-common terms of higher value (e.g., credit card numbers, SSN, uncommon names, etc.). In this embodiment, as in the previously abovementioned embodiment, the system avoids storing any copies of sensitive information by storing only hash codes and tuples of information related to the placement of cells in the database.

Detection of Preselected Data

[0080]   In one embodiment, the process of preselected data detection includes two major operations, or phases: indexing, and searching. In the indexing phase, the system builds indices from the preselected data. The preselected data may be any data whose relations allow it to be structured in a tabular format. In other words, the preselected data may be stored in a tabular format (e.g., data in a relational database, data in an Excel spreadsheet, etc.) or it may be stored in a non-tabular format but have such relationships as to allow it to be stored in a tabular format (e.g., data stored as comma separated values in a flat file or a password database, relational data in an object-oriented database, etc.).

[0081]   Figure 4 is a flow diagram of one embodiment of a process for indexing the preselected data. The process is performed by processing logic that may comprise hardware (circuitry, dedicated logic, etc.), software (such as is run on a general purpose computer system or a dedicated machine), or a combination of both.

[0082]   Referring to Figure 4, processing logic begins with determining whether the preselected data is stored in a standard tabular format (processing box 402). If not, processing logic converts the preselected data into a standard tabular format (processing

block 404). Each cell in the resulting table stores a fragment of the preselected data. In one embodiment, each data fragment is a token. A token may be a single word or a cluster of words (e.g., words enclosed in quotation marks). For example, while the word "this" may represent a token stored in a database cell, the phrase "this token" may also represent a standalone token if it is stored as a single string in a database cell.

[0083]   Next, processing logic creates a tuple-storage structure derived from the preselected data (processing block 406). A tuple-storage structure provides a mechanism for storing multiple tuples associated with the fragments of the preselected data. Examples of tuple-storage structures include a hash table, a vector, an array, a tree or a list. Each type of the tuple-storage structure is associated with a method for retrieving a set of tuples for any given content fragment (the set of tuples may be empty if no match is found in the tuple-storage structure).

[0084]   Further, processing logic stores information about the position of each data fragment within the database in a corresponding tuple (processing block 408). In one embodiment, the information about the position of a data fragment includes the number of a row storing the data fragment in the database. In another embodiment, this information also includes the number of a column storing the data fragment in the database and optionally the data type of the column.

[0085]   Afterwards, processing logic sorts the tuples in a predetermined order (e.g., in the ascending lexicographic order) (processing block 410).

[0086]   Thus, the resulting abstract data structure (i.e., the index) only contains information about the relative placement of data records in the context of the larger whole but does not include any fragments of the preselected data itself.

[0087]   In one embodiment, the contents of the index are treated cryptographically (e.g., with a hash function or using an encryption function with a cryptographic key) to further secure the index from theft.

[0088]   The "searching" phase of the preselected data detection process will now be discussed in more detail. Figure 5 is a flow diagram of one embodiment of a process for searching information content for preselected data. The process is performed by processing logic that may comprise hardware (circuitry, dedicated logic, etc.), software (such as is run on a general purpose computer system or a dedicated machine), or a combination of both.

[0089]   Referring to Figure 5, processing logic begins with receiving information content (processing block 502). The information content may be included in a file (e.g., an archived email message stored on a hard drive of a computer) or in a block of data transmitted over a network (e.g., an email message transmitted over a network using any type of a network protocol).

[0090]   Next, processing logic detects in the information content a sequence of content fragments that may possibly contain a portion of preselected data (processing block 504). As discussed above, the preselected data may be proprietary database data that needs to be protected or any other kind of data that has an inherent tabular structure. That is, the preselected data may either be stored in a tabular format (e.g., data in a relational database, data in an Excel spreadsheet, etc.) or it may be stored in a non-tabular format but have such relations as to allow it to be stored in a tabular format (e.g., data stored as comma separated values in a flat files or a password database, relational data in an object-oriented database, etc.).

[0091]   In one embodiment, the detected sequence of content fragments is a set of adjacent tokens within the information content. Each token may correspond to either a word or a phrase. The detected sequence of content fragments may be a portion of the received information content or the entire information content.

[0092]   In one embodiment, processing logic decides that a sequence of content fragments may possibly contain a portion of the preselected data upon determining that the sequence of content fragments resembles column-formatted data. This determination may be made by parsing the received information content to identify separated lines (as may be indicated, for example, by tags <cr> or <cr><lf>) and finding that these separated lines contain a similar number of tokens and optionally the similar data types of the tokens.

[0093]   In another embodiment, processing logic decides that a sequence of content fragments may possibly contain a portion of the preselected data upon parsing the entire information content and searching blocks of contiguous tokens for preselected data. In one embodiment, the blocks of contiguous tokens are defined based on user-specified parameters such as a user-specified width of each block and a user-specified position of each block within the information content (e.g., the user may require that the two adjacent blocks be separated by a certain number of tokens).

22

[0094]   In yet another embodiment, processing logic decides that a sequence of content fragments may possibly contain a portion of the preselected data upon finding in the information content an expression of a predefined format. Such expression may be, for example, an account number, a social security number, a credit card number, a phone number, a postal code, an email address, text formatting indicating a monetary or numeric value (e.g., "$" signs together with digits), etc. Once the expression is found, processing logic decides that a region of text surrounding the expression may possibly contain a portion of the preselected data. The size of this region may be defined by a predetermined number of tokens on each side of the found expression.

[0095]   In yet another embodiment, processing logic decides that a sequence of content fragments may possibly contain a portion of the preselected data upon determining that the word usage or the word distribution in the information content (or in some portion of the information content) resembles a statistical pattern that indicates a possible containment of the preselected data in the information content.

[0096]   In still another embodiment, processing logic decides that a sequence of content fragments may possibly contain a portion of the preselected data upon determining that certain properties associated with the received information content indicate a possible containment of the preselected data in the information content based on the history of previous violations. These properties may include, for example, the destination of the information content (e.g., a recipient of an electronic message), the origin of the information content, the time of transmission associated with the information content, the size of transmission associated with the information content, the types of files contained in the transmission (e.g., multipurpose Internet mail extension (MIME) types of files), etc. In one embodiment, the history of previous violations is maintained by identifying, for each detection of preselected data, the properties of the information content in which the preselected data was detected and recording these properties in a previous violation database. Subsequently, when processing logic decides whether a sequence of content fragments within the new information content may possibly contain a portion of preselected data, processing logic identifies the properties of the new information content and searches the previous violation database for these properties. If a match is found, processing logic determines whether the previous violations associated with the matching property indicate a possible containment of

preselected data in the new information content. This indication may be based on the number of previous violations associated with the matching property or the frequency of previous violations associated with the matching property. For example, this indication may be based upon the total number of violations that a particular sender has committed, or the frequency of those violations over a given time period.

[0097]   Afterwards, upon detecting a sequence of content fragments that may possibly contain a portion of the preselected data, processing logic makes a determination as to whether any subset of these content fragments matches a subset of the preselected data (processing block 506). This determination is made using an index (also referred to herein as an abstract data structure) that defines the tabular structure of the preselected data.

[0098]   Figure 6 is a flow diagram of one embodiment of a process for finding a match for a subset of content fragments in an abstract data structure derived from preselected data. The process is performed by processing logic that may comprise hardware (circuitry, dedicated logic, etc.), software (such as is run on a general purpose computer system or a dedicated machine), or a combination of both.

[0099]   Referring to Figure 6, processing logic begins with parsing the sequence of content fragments identified at processing block 504 of Figure 5 into content fragments (e.g., tokens). Then, for each content fragment, processing logic searches the abstract data structure for a set of matching tuples (processing block 602). For example, a word "Smith" contained in the information content may have several occurrences in the preselected data that are reflected in the abstract data structure. Specifically, each of these occurrences has a corresponding tuple in the abstract data structure. During the search, processing logic retrieves a set of tuples corresponding to the occurrences of the word "Smith" in the preselected data. Each tuple stores information about the position of this data fragment within a database or a table storing the preselected data. In one embodiment, the positional information includes the row number of a cell storing the data fragment. In another embodiment, the positional information also includes a column number of this cell and optionally the data type of the column.

[00100]   Next, processing logic combines the matching tuple sets found for all the content fragments (processing block 604) and then groups the combined matching tuple sets by row numbers into groups L (processing block 606). As a result, each group L

CA 02499508 2005-03-18

(referred to herein as an accumulator) contains matching tuple sets that all have the same column number, i.e., the matching tuple sets in each group L correspond to fragments of the preselected data that all appear to be from the same row in the database.

[00101]   Further, processing logic sorts the groups L by the number of matching tuple sets contained in each group (processing block 608) and, in one embodiment, selects those groups that have tuple sets with distinct column numbers (processing block 610). Afterwards, processing logic determines whether any of the selected groups has a sufficiently large number of matching tuple sets (processing block 612). For example, if the number of matching tuple sets in one group exceeds "3", then there is a high likelihood that the information content does include data from four or more columns of the same row in the database.

[00102]   Exemplary embodiments of a search process will now be described. Figures 7A – 7C are flow diagrams of alternate embodiments of a process for searching an incoming message using a hash table index of preselected data. The process is performed by processing logic that may comprise hardware (circuitry, dedicated logic, etc.), software (such as is run on a general purpose computer system or a dedicated machine), or a combination of both.

[00103]   Referring to Figure 7A, processing logic begins with parsing an incoming message (processing block 702). Next, processing logic determines whether the parsed portions of the incoming message contain column-formatted data (processing box 704). In one embodiment, lexical analysis may be used to identify lines in the parsed portions of the incoming message (e.g., by finding tags <cr> or <cr><lf> that are used to separate lines) and then detecting that the number of tokens found in adjacent lines is identical in number and in type. In one embodiment, processing logic stores the type of each token, along with the total number of tokens.

[00104]   If the determination made at processing box 704 is negative, processing transitions to processing block 702. Otherwise, processing transitions to processing block 706 where processing logic sets i equal to the first line that resembles column-formatted data.

[00105]   Next, processing logic applies a hash function H(k) to each token in line i (processing block 708), finds a set of tuples at H(k) in the hash table for each token in line i, adds the tuples to list L, and regroups list L into a set of accumulators (processing

block 712) in which each individual accumulator's tuples have the same row number value. Further, processing logic sorts that list L by the length of each A_i (processing block 714) and checks for unique occurrences of columns in sorted list L (processing block 716). At processing block 710, optional pre-processing logic may be performed to filter the tokens before insertion into list L so that only those tuples with type matching the lexical type of the original token k are added to L. It should be noted that in some other embodiments checking for unique occurrences of columns may be skipped for reasons of speed or simplicity. In yet other embodiments, tuples are simple "singletons" containing row numbers only (i.e., no column number and no type indicator.)

[00106]   Afterwards, if the incoming message contains more lines that resemble column-formatted data (processing box 718), processing logic increments i to the next line that resembles column-formatted data (processing block 722) and the process transitions to processing block 706. Otherwise, processing logic reports lines of text with A_i that exceed the predetermined size and have unique column numbers (processing block 720).

[00107]   Referring to Figure 7B, processing logic begins with receiving user-specified parameters of "width" (W) and "jump" (J) (processing block 732) and parsing an incoming message (processing block 734). Parameter W specifies the number of contiguous tokens in each block of contiguous tokens that is to be searched during a single iteration and parameter J specifies the required number of tokens between the two adjacent blocks.

[00108]   Next, processing logic sets the value of the location variable (S_t) to zero (processing block 736) and defines a block ("textblock") to be searched by collecting W contiguous tokens of the message starting at S_t (processing block 738).

[00109]   Further, processing logic applies a hash function H(k) to each token in the textblock (processing block 740), finds a set of tuples at H(k) in the hash table for each token in the textblock, adds the tuples that have the same type as the corresponding tokens in the textblock to list L (processing block 742), regroups list L into a set of accumulators (processing block 744), sorts that list L by the length of each A_i (processing block 746) and checks for unique occurrences of columns in sorted list L (processing block 748).

[00110]   Afterwards, processing logic increments S_t by J number of tokens (processing block 750) and determines whether location S_t is still within the message (processing box 752). If the determination is positive, the process transitions to processing block 738. Otherwise, processing logic reports textblocks with Ai that exceed the predetermined size and have unique column numbers (processing block 758).

[00111]   Referring to Figure 7C, processing logic begins with parsing an incoming message (processing block 764) and looking for a first expression having a user-specified format (processing block 766). Such expression may be, for example, an account number, a social security number, a credit card number, text formatting indicating a monetary or numeric value (e.g., "$" signs together with digits), etc. If the matching expression is not found, the process transitions to processing block 764. Otherwise, the process transitions to processing block 768 where processing logic defines a block ("textblock") to be searched by collecting W contiguous tokens before and after the matching expression. For example, the textblock may consist of 10 tokens immediately preceding the matching expression, the matching expression itself and 10 tokens immediately following the matching expression.

[00112]   Further, processing logic applies a hash function H(k) to each token in the textblock (processing block 770), finds a set of tuples at H(k) in the hash table for each token in the textblock, adds the tuples that have the same type as the corresponding tokens in the textblock to list L (processing block 772), regroups list L into a set of accumulators (processing block 774), sorts that list L by the length of each Ai (processing block 776) and checks for unique occurrences of columns in sorted list L (processing block 778).

[00113]   Afterwards, processing logic determines whether the message has anymore expressions of the user-specified format (processing box 780). If this determination is positive, the process transitions to processing block 768. Otherwise, processing logic reports textblocks with Ai that exceed the predetermined size and have unique column numbers (processing block 782).

An Exemplary Application

[00114]   In the normal course of operation in one embodiment, it is assumed that the PMS is positioned on a corporate network so that secure communications can occur with an organization's database (in which the records reside that require protection.) In the normal course of operations, it is also additionally assumed that the MMS is positioned so that it can monitor and/or intercept all outbound email communications of the organization.

[00115]   In this example, assume that the organization seeks protection on a database table called "CustomerRecords" that contains four columns: 1) first name, 2) last name, 3) credit card number, and 4) account balance. Employees of this organization would use the user interface application served by the PMS to specify that the CustomerRecords table required protection against theft via email. The PMS would then build an index of the records in the CustomerRecords table that consists of a hash table derived from the string values of the cells in the database. That is, the values in the cells are used to look values up in the hash table. The hash table itself contains records of the respective row number, column number, and data type of the cell itself. In the (frequent) cases of collisions in the hash table, a "collision list" holds multiple such records of row#, col #, and type. Once all cells in the database table are hashed into such a structure, the index is created and ready to be transmitted to the MMS. Note that the index contains no records of the database data itself. This is a key security constraint satisfied by this system.

[00116]   After the MMS receives the index, it parses the message and re-creates the hash table in memory in the same fashion as it was created in the PMS.

[00117]   As the MMS picks up outbound email messages and parses them, it uses this index in the manner described below to detect if any of these emails contain data from the database. This is done by parsing each individual line of text from the email messages. This may involve decoding the surrounding file types and converting everything into raw text (e.g., stripping all formatting information from a Microsoft Word file and leaving only the text itself.) This series of lines of text is then parsed into individual words by looking for separation marks like the "space" character, or other forms of punctuation. These words are text tokens. For each line of text tokens, this

system then consults the index by applying the hash function to each token. The result of this operation is a hash table collision list for each token on the line. As described above, each collision list is itself a set of data elements that store possible row number, column number, and type triplets. If the union of all triplets from all collision lists is taken, and if a set of triplets is found with all with the same row number, but with distinct column numbers, then with high probability this line of text from the email message contains a record from the database. It should be noted that the term "tuple" used herein is not limited to the specific case of the triplets of row number, column number, and type and may refer to data structures that do not contain all of these three parameters. For example, in one embodiment, a tuple contains the row number but not the column number and the type of the database data.

Comparison with the Prior Art

[00118]   Database query mechanisms are significantly different from the teachings described herein. One difference is that B-trees actually contain fragments of the database tables that they index. In the approach described above, there are no copies of the database data stored inside the index. The reason that this is important is that – as mentioned above – the MMS has to have a copy of the index in order to protect the data from escape; however the MMS is also best deployed in a position in the network where it may be exposed to significant threats. Keeping the index that the MMS uses free of any components of the database data is a key requirement.

[00119]   Another difference between standard database query mechanisms and the invention outlined here has to do with the types of queries that are required. The standard set of queries used in relational databases is based on predicate logic using connectives like AND and OR. This basic system does not work well for detection of database data that is typically cut-and-paste into email and webmail messages. Database data that is cut-and-paste into email messages is typically from reports and will often contain data in each line that is extraneous and not found inside the database table. An example could be an email message that contains, for example, account information for a bunch of customers. Such a message will contain plenty of records from the core database that requires protection, e.g., first name, last name, social-security number, etc.,

29

but could also contain information not in the core database tables. A typical example is information that is "joined" from other databases. Another example is simple line formatting tokens that separate fields of database data. Because of the possibility of this extra data that's typically found on each of these lines, the standard predicate logic connectives like AND and OR applied to each token on the line of an outgoing message produce either too many hits (as is the case with OR) or zero hits (as is the case with AND). In the description herein, the system is able to detect the presence of n or more tokens that are all from the same row of a database table, even in the case where n is much smaller than the total number of tokens in the line. This is another significant difference between the present invention and the prior art mentioned above for database and document query mechanisms.

[00120]   There are several major differences between the techniques described above and information retrieval technologies. Firstly, the indices for these systems contain (inside the concordances) the same terms that are stored in the database that is to be protected. Here again, since the system deploys this index into a location on the network that is potentially under hacker threat; this is a definite disadvantage. Secondly, these query systems run Boolean queries using the forms of predicate logic like AND and OR. As mentioned above, this approach is at a distinct disadvantage for detecting database records that have been possibly "joined" with extraneous data from other tables.

[00121]   The technique of file shingling is similar to, but substantially different from the technique described herein. In file shingling, the subject of interest is text data (prose, software, outlines, etc.). In the techniques described here, the focus is on protecting database data. One difference is that database data from a given database table may appear with the row order or column order permuted arbitrarily in the test message. These permutations are the simple result of the query mechanisms typically applied to extract database data. A database query could result in a block of database data that comes in arbitrary column order, and arbitrary row order. For this reason, the basic technique of file shingling will not work if applied to database data. File shingling assumes that the same linear sequence is followed between the protected document and the test document.

[00122]   There are many important differences between internet content filtering systems and the teachings described herein. As mentioned above, Internet content

CA 02499508 2005-03-18

filtering systems are based on keyword searches. The novel techniques described above build an abstract data structure from the database data that it seeks to protect. This abstract data structure does not contain fragments of the text it is trying to protect. A keyword filtering system must contain some representation of the text that it is searching for in order to run its queries. The second major difference is that these Internet content filtering systems are not intended to protect database data. Using regular expression matching to detect violations of an organizations privacy policy on database data will also lead to a very inaccurate method of detection. These systems are primarily applied to stop employee abuse of the Internet as it relates to pornographic or abusive content and language. Such systems, if applied to the protection of database data, would use regular expressions to match database records. This would also result in transferring fragments of the database data to the computer on the network where security risks are maximized.

An Exemplary Computer System

[00123]   Figure 8 is a block diagram of an exemplary computer system that may perform one or more of the operations described herein. Referring to Figure 8, computer system 800 may comprise an exemplary client 850 or server 800 computer system. Computer system 800 comprises a communication mechanism or bus 811 for communicating information, and a processor 812 coupled with bus 811 for processing information. Processor 812 includes a microprocessor, but is not limited to a microprocessor, such as, for example, Pentium^{TM}, PowerPC^{TM}, Alpha^{TM}, etc.

[00124]   System 800 further comprises a random access memory (RAM), or other dynamic storage device 804 (referred to as main memory) coupled to bus 811 for storing information and instructions to be executed by processor 812. Main memory 804 also may be used for storing temporary variables or other intermediate information during execution of instructions by processor 812.

[00125]   Computer system 800 also comprises a read only memory (ROM) and/or other static storage device 806 coupled to bus 811 for storing static information and instructions for processor 812, and a data storage device 807, such as a magnetic disk or optical disk and its corresponding disk drive. Data storage device 807 is coupled to bus 811 for storing information and instructions.

31

[00126]   Computer system 800 may further be coupled to a display device 821, such as a cathode ray tube (CRT) or liquid crystal display (LCD), coupled to bus 811 for displaying information to a computer user. An alphanumeric input device 822, including alphanumeric and other keys, may also be coupled to bus 811 for communicating information and command selections to processor 812. An additional user input device is cursor control 823, such as a mouse, trackball, trackpad, stylus, or cursor direction keys, coupled to bus 811 for communicating direction information and command selections to processor 812, and for controlling cursor movement on display 821.

[00127]   Another device that may be coupled to bus 811 is hard copy device 824, which may be used for printing instructions, data, or other information on a medium such as paper, film, or similar types of media. Furthermore, a sound recording and playback device, such as a speaker and/or microphone may optionally be coupled to bus 811 for audio interfacing with computer system 800. Another device that may be coupled to bus 811 is a wired/wireless communication capability 825 to communication to a phone or handheld palm device.

[00128]   Note that any or all of the components of system 800 and associated hardware may be used in the present invention. However, it can be appreciated that other configurations of the computer system may include some or all of the devices.

[00129]   Whereas many alterations and modifications of the present invention will no doubt become apparent to a person of ordinary skill in the art after having read the foregoing description, it is to be understood that any particular embodiment shown and described by way of illustration is in no way intended to be considered limiting. Therefore, references to details of various embodiments are not intended to limit the scope of the claims which in themselves recite only those features regarded as essential to the invention.

CLAIMS

We claim:

1.   A method comprising:

monitoring messages electronically transmitted over a network for embedded preselected data; and

performing content searches on the messages to detect the presence of the embedded preselected data using an abstract data structure derived from the preselected data.

2.   The method defined in Claim 1 wherein the preselected data comprises database data.

3.   The method defined in Claim 1 wherein the abstract data structure comprises an index.

4.   The method defined in Claim 3 wherein the index does not include a copy of the preselected data.

5.   The method defined in Claim 3 wherein the index comprises a relative placement of elements in a database in relation to other elements in the database.

6.   The method defined in Claim 3 wherein the index comprises a hash table derived from string values of the cells of database data.

7.   The method defined in Claim 3 wherein the index comprises a row number, column number and type of column associated with a fragment within the database.

8.   The method defined in Claim 1 further comprising:

querying a database;

extracting at least one copy of the preselected data from the database; and

creating the abstract data structure based on the preselected data extracted from the database.

9.   The method defined in Claim 1 further comprising creating the abstract data structure based on the preselected data extracted from a database.

10.   The method defined in Claim 9 wherein creating the abstract data structure comprises:

storing a row number, column number, and data type indicator for each cell in a database table into a hash table; and

sorting the hash table based on a predefined order into a collision list.

11.   The method defined in Claim 10 wherein the order comprises ascending lexicographic order.

12.   The method defined in Claim 1 wherein performing content searches on the messages using an abstract data structure derived from the preselected data comprises:

parsing individual lines of text in one of the messages to parse the one message into individual words;

applying a hash function to each of the individual words in a line to generate individual hash table collision lists for each of the individual words on each of the individual lines; and

identifying data sets within the individual hash table collision lists with common row numbers and distinct column numbers as preselected data.

13.   The method defined in Claim 1 wherein the preselected data is selected based on a policy.

14. The method defined in Claim 1 further comprising accepting user input specifying the policy.

15. The method defined in Claim 14 wherein accepting user input specifying the policy comprises receiving information on a graphical user interface.

16. The architecture defined in Claim 13 wherein the policy includes a specification of the data and a network location of the data.

17. The architecture defined in Claim 16 wherein the specification of the data comprises a table name.

18. The architecture defined in Claim 16 wherein the specification of the data comprises a database identifier.

19. The architecture defined in Claim 16 wherein the network location comprises an IP address of a server.

20. The architecture defined in Claim 19 wherein the network location comprises a server identifier.

21. The method defined in Claim 1 wherein monitoring messages occurs at a plurality of exit points of the network.

22. The method defined in Claim 1 further comprising:
periodically querying a database;
extracting copies of data base data that is to be protected; and
deriving an abstract data structure based on extracted data and its location in the database.

23. The method defined in Claim 1 further comprising sending the abstract data structure to a message monitoring system.

24.   The method defined in Claim 1 further comprising:
preventing escape of messages containing preselected database content.

25.   The method defined in Claim 1 further comprising:
logging messages that contain database content.

26.   The method defined in Claim 1 further comprising:
reporting violations of the policy.

27.   The method defined in Claim 1 further comprising:
intercepting one or more messages; and
re-routing one or more messages to a new destination.

28.   An architecture comprising:
a policy management system to set a policy; and
a message monitoring system to implement the policy by
          monitoring messages electronically transmitted over a network for
embedded preselected data; and
          performing content searches on the messages to detect the presence of the
embedded preselected data using an abstract data structure derived from the preselected
data.

29.   The architecture defined in Claim 28 wherein the policy includes a
specification of the data and a network location of the data.

30.   The architecture defined in Claim 29 wherein the specification of the data
comprises a table name.

31.   The architecture defined in Claim 29 wherein the specification of the data
comprises a database identifier.

32.   The architecture defined in Claim 29 wherein the network location comprises an IP address of a server.

33.   The architecture defined in Claim 32 wherein the network location comprises a server identifier.

34.   The architecture defined in Claim 32 wherein the network location comprises a file name of columnar-formatted data.

35.   The architecture defined in Claim 28 wherein the policy management system and the message monitoring system are incorporated into the same physical system.

36.   The architecture defined in Claim 28 wherein the policy management system and the message monitoring system are incorporated into the same logical system.

37.   The architecture defined in Claim 28 wherein the policy management system and the message monitoring system do not reside on the same local area network (LAN).

38.   The architecture defined in Claim 28 wherein the policy management system and the message monitoring system reside on two distinct LANs coupled together via the Internet.

39.   A method comprising:
receiving information content;
detecting, in the information content, a sequence of content fragments that may contain a portion of preselected data; and
determining whether a subset of content fragments within the sequence matches any sub-set of the preselected data using an abstract data structure that defines a tabular structure of the preselected data.

40.   The method of claim 39 wherein the pre-selected data is stored in at least one of an Excel spreadsheet, a flat file, and a database.

41.   The method of claim 39 wherein each content fragment corresponds to either a word or a phrase.

42.   The method of claim 39 wherein the sequence of content fragments is a set of adjacent tokens within the information content included in a file.

43.   The method of claim 39 wherein the sequence of content fragments is a set of adjacent tokens within the information content included in a block of data transmitted over a network.

44.   The method of claim 39 wherein detecting a sequence of content fragments that may contain a portion of pre-selected data comprises:

        identifying lines within the information content; and

        determining that sequences of content fragments within the lines have a similar number of content fragments and similar data types of content fragments.

45.   The method of claim 39 wherein detecting a sequence of content fragments that may contain a portion of pre-selected data comprises:

        searching the information content for an expression having a predefined format; and

        deciding that a region surrounding the expression may contain a portion of pre-selected data.

46.   The method of claim 45 wherein the expression having a predefined format is any one of an account number, a social security number, a credit card number, a phone number, a postal code, an email address, a number, a currency amount, or a driver's license number.

47.   The method of claim 39 wherein detecting a sequence of content fragments that may contain a portion of pre-selected data comprises:

finding, within the information content, the sequence of content fragments that resembles a predefined statistical pattern.

48.   The method of claim 47 wherein resemblance to the predefined statistical pattern is based either on word distribution or word usage.

49.   The method of claim 39 wherein detecting a sequence of content fragments that may contain a portion of pre-selected data comprises:

determining that one or more properties associated with the information content indicate a possible containment of the preselected data in the information content based on history of previous violations.

50.   The method of claim 49 wherein the one or more properties are selected from the group consisting of destination of the information content, origin of the information content, time of transmission associated with the information content, size of transmission associated with the information content, and types of files contained within the transmission.

51.   The method of claim 39 wherein the abstract data structure comprises a tuple-storage structure derived from the preselected data.

52.   The method of claim 51 wherein the abstract data structure comprises a plurality of tuples, each of the plurality of tuples including a row numbers of a data item in a corresponding cell of the tabular structure of the preselected data.

53.   The method of claim 52 wherein each of the plurality of tuples additionally includes a column number and optionally a column type of the data item in the corresponding cell.

54.   The method of claim 53 wherein determining whether a sub-set of the content fragments within the sequence matches any sub-set of the pre-selected data comprises:

finding a set of matching tuples in the abstract data structure for each content fragment in the sequence;

combining sets of matching tuples found for all content fragments in the sequence; and

grouping the combined sets of matching tuples by row numbers into groups of matching tuple sets.

55.   The method of claim 54 wherein determining whether a sub-set of the content fragments within the sequence matches any sub-set of the pre-selected data further comprises:

sorting the groups of matching tuple sets by the number of matching tuple sets contained in each group;

selecting groups that have matching tuple sets with distinct column numbers; and

determining whether any of the selected groups have a number of matching tuple sets that exceeds a predefined threshold.

56.   The method of claim 39 wherein the content of the abstract data structure is treated cryptographically using any one of a hash function or an encryption function with a cryptographic key.

57.   An apparatus comprising:

means for receiving information content;

means for detecting, in the information content, a sequence of content fragments that may contain a portion of preselected data; and

means for determining whether a subset of content fragments within the sequence matches any sub-set of the preselected data using an abstract data structure that defines a tabular structure of the preselected data.

58.   A system comprising:

40

a memory containing an abstract data structure that defines a tabular structure of preselected data; and

at least one processor coupled to the memory, the at least one processor executing a set of instructions which cause the processor to receive information content, detect, in the information content, a sequence of content fragments that may contain a portion of the preselected data, and determine whether a subset of content fragments within the sequence matches any sub-set of the preselected data using the abstract data structure.

59.   A computer readable medium that provides instructions, which when executed on a processor cause the processor to perform a method comprising:

receiving information content;

detecting, in the information content, a sequence of content fragments that may contain a portion of preselected data; and

determining whether a subset of content fragments within the sequence matches any sub-set of the preselected data using an abstract data structure that defines a tabular structure of the preselected data.

60.   A method comprising:

searching contents of a plurality of data storage media of a personal computing device for pre-selected sensitive data; and

if at least a portion of the pre-selected sensitive data is detected, sending a notification of detection of the pre-selected sensitive data to a system via a network.

61.   The method of claim 60 further comprising:

if at least a portion of the pre-selected sensitive data is detected, preventing access to the detected data.

62.   The method of claim 60 wherein the content is searched periodically.

63.   The method of claim 60 wherein the content is searched when the personal computing device is disconnected from the network.

41

64.   The method of claim 63 wherein sending a notification comprises:

upon detecting the pre-selected sensitive data, creating a message containing the notification of the detection of the pre-selected sensitive data;

placing the message in a transmission queue; and

transmitting the message to the system after the personal computing device is re-connected to the system.

65.   The method of claim 60 further comprising:

receiving instructions defining a scope of a search for the personal computing device from the system.

66.   The method of claim 60 wherein searching contents of a plurality of data storage media within a personal computing device comprises:

receiving an abstract data structure associated with the pre-selected sensitive data; and

utilizing the abstract data structure when searching the contents of the plurality of data storage media of the personal computing device for the pre-selected sensitive data.

67.   The method of claim 60 wherein searching contents of a plurality of data storage media of the personal computing device comprises monitoring one or more specific data operations for presence of at least a portion of the pre-selected sensitive data.

68.   The method of claim 67 wherein at least one of the one or more specific data operations is selected from the group consisting of a file-read, a file-write, a file-update, a read from a removable media device, a write to a removable media device, and access of data stored on any of the plurality of data storage media by a program running on the personal computing device.

69.   The method of claim 60 wherein the pre-selected sensitive data has a tabular format.

70.   The method of claim 60 wherein the pre-selected sensitive data is capable of being re-structured into a tabular format based on relationships among elements of the pre-selected sensitive data.

71.   The method of claim 60 wherein the pre-selected sensitive data is maintained by an organization in at least one of a spreadsheet, a flat file, and a database.

72.   The method of claim 71 wherein the abstract data structure comprises a tuple-storage structure derived from the pre-selected data.

73.   The method of claim 72 wherein the abstract data structure comprises a plurality of tuples, each of the plurality of tuples including a row numbers of a data item in a corresponding cell of a tabular structure of the pre-selected data.

74.   The method of claim 73 wherein each of the plurality of tuples additionally includes a column number and optionally a column type of the data item in the corresponding cell.

75.   The method of claim 60 wherein the plurality of data storage media is selected from the group consisting of a main memory, a static memory, and a mass storage memory.

76.   The method of claim 60 wherein searching contents of a plurality of data storage media comprises:
        searching content of each volatile storage device within the plurality of data storage media; and
        searching content of each persistent storage device within the plurality of data storage media.

77.   The method of claim 76 further comprising detecting use of the pre-selected data by an application running on the personal computing device.

43

78.   The method of claim 76 further comprising:

identifying the application using the pre-selected data; and

reporting the identified application.

79.   An apparatus comprising:

means for searching contents of a plurality of data storage media of a personal
computing device for pre-selected sensitive data; and

means for sending a notification of detection of the pre-selected sensitive data to
a system via a network if at least a portion of the pre-selected sensitive data is detected.

80.   The apparatus of claim 79 wherein the content is searched periodically.

81.   The apparatus of claim 79 wherein the content is searched when the
personal computing device is disconnected from the network.

82.   The apparatus of claim 79 wherein means for sending a notification
comprises:

means for creating a message containing the notification of the detection of the
pre-selected sensitive data upon detecting the pre-selected sensitive data;

means for placing the message in a transmission queue; and

means for transmitting the message to the system after the personal computing
device is re-connected to the system.

83.   The apparatus of claim 79 further comprising:

means for receiving instructions defining a scope of a search for the personal
computing device from the system.

84.   The apparatus of claim 79 wherein means for searching contents of a
plurality of data storage media of the personal computing device comprises means for
monitoring one or more specific data operations for presence of at least a portion of the
pre-selected sensitive data.

85. The apparatus of claim 84 wherein at least one of the one or more specific data operations is selected from the group consisting of a file-read, a file-write, a file-update, a read from a removable media device, a write to a removable media device, and access of data stored on any of the plurality of data storage media by a program running on the personal computing device.

86. The apparatus of claim 79 wherein the plurality of data storage media is selected from the group consisting of a main memory, a static memory, and a mass storage memory.

87. The apparatus of claim 79 wherein means for searching contents of a plurality of data storage media comprises:

means for searching content of each volatile storage device within the plurality of data storage media; and

means for searching content of each persistent storage device within the plurality of data storage media.

88. The apparatus of claim 87 further comprising means for detecting use of the pre-selected data by an application running on the personal computing device.

89. The apparatus of claim 87 further comprising:

means for identifying the application using the pre-selected data; and

means for reporting the identified application.

90. A personal computing device comprising:

a plurality of storage media storing various data; and

at least one processor coupled to the plurality of storage media, at least one processor executing a set of instructions which cause the processor to search contents of the plurality of data storage media for pre-selected sensitive data, and to send a notification of detection of the pre-selected sensitive data to a system via a network if at least a portion of the pre-selected sensitive data is detected.

45

91.   A computer readable medium that provides instructions, which when executed on a processor cause the processor to perform a method comprising:

searching contents of a plurality of data storage media of a personal computing device for pre-selected sensitive data; and

if at least a portion of the pre-selected sensitive data is detected, sending a notification of detection of the pre-selected sensitive data to a server via a network.

1/11

①	②	③	④
Identify highest-value information	**Author policies**	**Surveillance & Enforcement**	**Actionable business intelligence**
• Information – Customer data – Financial data – Product plan info • Stored in... – DB tables – DB-based apps	• WHO can take action on information – "Contractors" – "Executives" – "All employees" • WHAT they can do – E-mail – HTTP post – IM – Print – Floppy/CD • HOW to respond to violation	• Monitor network traffic and employee actions • Real-time review of content • Policy and strength of match drives enforcement • Soft policy: log incidents, track data, real-time alerts • Hard policy: also block transaction	• Compliance reports • Do wrong people have access to information • Damage control for specific incidents • Specific management actions • Feedback loop for effectiveness of policies

FIG. 1

Surveillance Mode

FIG. 2A

Hard Enforcement Mode

FIG. 2B

2/11

BEGIN

301 — MONITO R MESSAGES FOR
PRES ELECTED DATA

302 — MESSAGES WITH
PRESELECTED D ATA ?   NO

YES

303 — IS INDIVIDUAL
SE NDING/ RECEIVI NG
MESSAG ES AUTHORIZED TO
SEND/RE CEIVE THIS
INFORMATION ?   NO

YES

304 — TAKE ACTION (E. G., LOG
MESSAGE, RE-RO UTE
MESSAGE, INTERCEPT
MESSAGE, ET C.)

FIG. 3

CA 02499508 2005-03-18

3/11

```
              ┌─────────┐
              │  START  │
              └─────────┘
                   │
                   │        ┌─402
                   ▼
              ╱─────────────╲
         Y   ╱  PRESELECTED  ╲
    ◄───────┤   DATA STORED IN │
            ╲   A TABULAR     ╱
             ╲  FORMAT?      ╱
              ╲─────────────╱
                   │
                   │ N         ┌─404
                   ▼
    ┌──────────────────────────────────┐
    │  CONVERT PRESELECTED    DATA      │
    │     INTO TABULAR FORMAT           │
    └──────────────────────────────────┘
                   │                ┌─406
                   ▼
    ┌──────────────────────────────────┐
    │  CREATE A TUPLE-STORAGE STRUCTURE │
    │  DERIVED FROM THE PRESELECTED DATA│
    └──────────────────────────────────┘
                   │                ┌─408
                   ▼
    ┌──────────────────────────────────┐
    │  STORE INFORMATION ABOUT RELATIVE │
    │ PLACEMENT OF EACH DATA FRAGMENT WITHIN │
    │ THE DATABASE IN A CORRESPONDING TUPLE │
    └──────────────────────────────────┘
                   │                ┌─410
                   ▼
    ┌──────────────────────────────────┐
    │ SORT THE TUPLES IN A PREDETERMINED ORDER │
    │   (E.G. ASCENDING LEXICOGRAPHIC ORDER)   │
    └──────────────────────────────────┘
                   │
                   ▼
              ┌─────────┐
              │   END   │
              └─────────┘
```

FIG. 4

4/11

START

502

RECEIVE INFORMATION   CONTENT

504

DETECT, IN THE INFORMATION CONTENT, A
SEQUENCE OF CONTENT FRAGMENTS THAT M   AY
CONTAIN A P ORTION OF PRESELECTED DATA

506

DETERMINE WHETHER A SUBSET OF THE
CONTENT FRAGMENTS MATCHES ANY   SUBSET
OF THE PRESELECTED DATA USING AN INDEX
OF THE PRESELECTED   DATA

END

FIG. 5

5/11

```
                        ┌─────────┐
                        │  START  │
                        └─────────┘
                             │
                             ▼                        ┌ 602
            ┌──────────────────────────────────┐
            │   FIND A SET OF M ATCHING TUPLES IN │
            │   THE ABSTRACT DATA  STRUCTURE FOR  │
            │   EACH CONTENT FR AGMENT IN THE     │
            │        TARGETED POR TION            │
            └──────────────────────────────────┘
                             │
                             ▼                        ┌ 604
            ┌──────────────────────────────────┐
            │  COMBI NE SETS OF MATCHING TUPLES FOUND │
            │      FOR ALL  CONTENT FRAGMENTS     │
            └──────────────────────────────────┘
                             │
                             ▼                        ┌ 606
            ┌──────────────────────────────────┐
            │   GROUP  THE COMBINED SETS OF MATCHING │
            │   TUPLES BY ROW NUMBERS IN    TO GROUPS │
            │        OF MATCHING TUPLES SETS      │
            └──────────────────────────────────┘
                             │
                             ▼                        ┌ 608
            ┌──────────────────────────────────┐
            │  SORT THE  GROUPS OF MATCHING TUPLE SETS │
            │   BY THE NUMBER OF MATCHING TUPLE SE   TS │
            │        CONTAINED  IN EACH GROUP     │
            └──────────────────────────────────┘
                             │
                             ▼                        ┌ 610
            ┌──────────────────────────────────┐
            │  SELECT G ROUPS THAT HAVE MATCHING TUPLE │
            │    SETS WITH DISTINCT COLU   MN NUMBERS  │
            └──────────────────────────────────┘
                             │
                             ▼                        ┌ 612
            ┌──────────────────────────────────┐
            │    DET ERMINE WHETHER ANY OF THE    │
            │   SELECTED  GROUPS HAVE A NUMBER OF │
            │    MATCH NG T UPLE SETS THAT EXCEEDS │
            │        A PREDEFINED THRESHOLD       │
            └──────────────────────────────────┘
                             │
                             ▼
                        ┌─────────┐
                        │   END   │
                        └─────────┘
```

FIG. 6

```
                        ┌─────────┐
                        │  START  │
                        └─────────┘
                             │
702 ──┐   ┌──────────────────▼──────────────────┐
      └──▶│         PARSE MESSAGE                │
          └─────────────────┬───────────────────┘
                            │
704 ──┐          ┌──────────▼──────────┐    NO
      └──▶        IS THERE COLUMN-       ─────────┐
                 FORMATTED DATA ?                 │
                 └──────────┬──────────┘          │
                         YES│                     │
          ┌─────────────────▼───────────────┐     │
          │  SET i EQUAL TO 1ST LINE THAT    │  706│
          │  RESEMBLES COLUMN-FORMATTED      │     │
          │  DATA                            │     │
          └─────────────────┬───────────────┘     │
                            │◄─────────────────────────┐
          ┌─────────────────▼───────────────┐  708     │
          │  APPLY HASH FUNCTION H (k) TO    │          │
          │  EACH TOKEN IN LINE i            │          │
          └─────────────────┬───────────────┘          │
                            │                           │
          ┌─────────────────▼───────────────┐  710      │
          │  ADD TUPLES FROM HASH TABLE AT   │          │
          │  H(k) THAT HAVE THE SAME TYPE AS │          │
          │  TOKENS IN LINE i TO LIST L      │          │
          └─────────────────┬───────────────┘          │
                            │                           │
          ┌─────────────────▼───────────────┐  712      │
          │  REGROUP L INTO                  │          │
          │  SET OF ACCUMULATORS (A1, A2, A3,│          │
          │  ..., An) WHERE A i IS LIST OF ALL│         │
          │  ELEMENTS IN L CORRESPONDING TO A│          │
          │  UNIQUE ROW NUMBER               │          │
          └─────────────────┬───────────────┘          │
                            │                           │
          ┌─────────────────▼───────────────┐  714      │
          │  SORT LIST L BY                  │          │
          │  LENGTH OF EACH Ai               │          │
          └─────────────────┬───────────────┘          │
                            │                  716      │
          ┌─────────────────▼───────────────┐      ┌─────────────────────┐ 722
          │  CHECK FOR UNIQUE OCCURRENCES OF │      │ INCREMENT i TO NEXT  │
          │  COLUMNS IN SORTED LIST L        │      │ LINE THAT RESEMBLES  │
          └─────────────────┬───────────────┘      │ COLUMN-FORMATTED DATA│
                            │          718          └──────────▲──────────┘
                 ┌──────────▼──────────┐                       │
                  MORE LINES THAT              YES              │
                 RESEMBLE COLUMN     ──────────────────────────┘
                 FORMATTED DATA?
                 └──────────┬──────────┘
                         NO │
          ┌─────────────────▼───────────────┐  720
          │  REPORT LINES OF TEXT WITH       │
          │  Ai THAT EXCEED PREDETERMINED    │
          │  SIZE AND HAVE UNIQUE            │
          │  COLUMN NUMBERS                  │
          └─────────────────┬───────────────┘
                            │
                        ┌───▼─────┐
                        │  START  │
                        └─────────┘
```

FIG. 7A

7/11

```
                        ┌──────────┐
                        │  START   │
                        └──────────┘
                              │
                              ▼
        ┌───────────────────────────────────┐
        │     RECEIVE USER-SPECIFIED         │───  732
        │  PARAMETERS OF "WIDTH" (W) AND "   │
        │            JUMP" (J)               │
        └───────────────────────────────────┘
                              │
                              ▼
        ┌───────────────────────────────────┐
        │                                   │───  734
        │     PARSE AN INCOMING MESSAGE     │
        │                                   │
        └───────────────────────────────────┘
                              │
                              ▼
        ┌───────────────────────────────────┐
        │   SET THE VALUE OF THE LOCATION   │───  736
        │      VARIABLE (Sᵢ) TO ZERO        │
        └───────────────────────────────────┘
                              │
                              ▼
        ┌───────────────────────────────────┐
        │ DEFINE A BLOCK ("TEXTBLOCK") TO   │───  738
        │  BE SEARCHED BY COLLECTING W      │
        │   CONTIGUOUS TOKENS OF THE        │
        │    MESSAGE STARTING AT Sᵢ         │
        └───────────────────────────────────┘
                              │
                              ▼
        ┌───────────────────────────────────┐
        │  APPLY HASH FUNCTION H (k) TO EACH │───  740
        │     TOKEN IN THE TEXTBLOCK        │
        └───────────────────────────────────┘
                              │
                              ▼
        ┌───────────────────────────────────┐
        │  ADD TUPLES FROM HASH TABLE AT    │───  742
        │ H(k) THAT HAVE THE SAME TYPE AS   │
        │ TOKENS IN THE TEXTBLOCK TO LIST L │
        └───────────────────────────────────┘
                              │
                              ▼
        ┌───────────────────────────────────┐
        │        REGROUP L INTO             │───  744
        │  SET OF ACCUMULATORS {A1, A2, A3, │
        │  ..., An) WHERE Ai IS LIST OF ALL │
        │ ELEMENTS IN L CORRESPONDING TO A  │
        │      UNIQUE ROW NUMBER            │
        └───────────────────────────────────┘
                              │
                              ▼
        ┌───────────────────────────────────┐
        │        SORT LIST L BY             │───  746
        │      LENGTH OF EACH Ai            │
        └───────────────────────────────────┘
                              │
                              ▼
        ┌───────────────────────────────────┐
        │ CHECK FOR UNIQUE OCCURRENCES OF   │───  748
        │   COLUMNS IN SORTED LIST L        │
        └───────────────────────────────────┘
                              │
                              ▼
        ┌───────────────────────────────────┐
        │      INCREMENT Sᵢ BY J            │───  750
        │     NUMBER OF TOKENS              │
        └───────────────────────────────────┘
                              │
                              ▼         752
                         ◇───────────◇
                        ╱  LOCATION   ╲      YES
                       ╱  Sᵢ WITHIN THE ╲ ──────────
                       ╲   MESSAGE?    ╱
                        ╲─────────────╱
                              │
                             NO
                              ▼
        ┌───────────────────────────────────┐
        │ REPORT TEXTBLOCKS WITH Ai THAT    │───  754
        │ EXCEED THE PREDETERMINED SIZE     │
        │  AND HAVE UNIQUE COLUMN           │
        │        NUMBERS                    │
        └───────────────────────────────────┘
                              │
                              ▼
                        ┌──────────┐
                        │   END    │    FIG. 7B
                        └──────────┘
```

8/11

```
                    ( START )
                        |
                        v
764 ───┌─────────────────────────────┐
        │  PARSE AN INCOMING MESSAGE   │
        └─────────────────────────────┘
                        |
                        v
               ╱  FIRST       ╲
766 ───      ╱ EXPRESSION HAVING A ╲        NO
            ╱  USER-SPECIFIED      ╲ ──────────┐
            ╲    FORMAT IS         ╱            │
             ╲     FOUND?        ╱              │
                   YES                          │
                    |                           │
                    v                           │
      ┌─────────────────────────────┐           │
      │ DEFINE A BLOCK ("TEXTBLOCK") TO │        │
      │   BE SEARCHED BY COLLECTING   │ ─ 768   │
      │      W CONTIGUOU S TOKENS     │          │
      │     BEFORE AND AFTER THE      │          │
      │     MATCHING EXPRE SSION      │          │
      └─────────────────────────────┘           │
                    |                            │
                    v                            │
      ┌─────────────────────────────┐   770     │
      │  APPLY HASH FUNCTION H (k) TO EACH │      │
      │     TOKEN IN THE TEXTBLOCK    │          │
      └─────────────────────────────┘           │
                    |                            │
                    v                            │
      ┌─────────────────────────────┐   772     │
      │ ADD TUPLES FROM HASH TABLE  AT │          │
      │ H(k) THAT HAVE THE SAME TYPE AS │        │
      │ TOKENS IN THE TEXTBLOCK TO LIST L │      │
      └─────────────────────────────┘           │
                    |                            │
                    v                            │
      ┌─────────────────────────────┐            │
      │     REGROUP L INTO           │           │
      │ SET OF ACCUMULATORS {A1, A2, A3, │ 774   │
      │ ..., An} WHERE Ai IS LIST OF ALL │       │
      │ ELEMENTS IN L CORRESPONDING TO A │       │
      │    UNIQUE ROW NUMBER          │          │
      └─────────────────────────────┘           │
                    |                            │
                    v                            │
      ┌─────────────────────────────┐   776     │
      │       SORT LIST L BY          │          │
      │    THE LENGTH OF EACH  Ai     │          │
      └─────────────────────────────┘           │
                    |                            │
                    v                            │
      ┌─────────────────────────────┐   778     │
      │ CHECK FOR UNIQUE OCCURRENCES OF │        │
      │   COLUMNS IN SORTED LIST L    │          │
      └─────────────────────────────┘           │
                    |          780               │
                    v                            │
               ╱  MORE       ╲                   │
              ╱ EXPRESSIONS OF USER- ╲    YES     │
             ╱  SPECIFIED FORMAT IN   ╲ ─────────┘
             ╲    MESSAGE?          ╱
                   NO
                    |
                    v
      ┌─────────────────────────────┐
      │ REPORT TEXTBLOCK S WITH      │
      │ Ai THAT EXCEED PREDETERMINED │ ─ 782
      │   SIZE AND HAVE UN IQUE      │
      │    COLUMN NUMBERS            │
      └─────────────────────────────┘
                    |
                    v
                 ( END )            FIG. 7C
```

800

```
┌──────────────┐  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐
│    MAIN      │  │   STATIC     │  │    MASS      │  │              │
│   MEMORY     │  │   MEMORY     │  │  STORAGE     │  │ PROCESS OR   │
│    804       │  │    806       │  │  MEMORY      │  │    812       │
│              │  │              │  │    807       │  │              │
└──────────────┘  └──────────────┘  └──────────────┘  └──────────────┘
        ↕                ↕                 ↕                 ↕
┌─────────────────────────────────────────────────────────────────────┐
│                              BUS                                      │
│                              811                                      │
└─────────────────────────────────────────────────────────────────────┘
                                  ↕
┌──────────┐  ┌──────────┐  ┌──────────┐  ┌──────────┐  ┌──────────────┐
│ DISPLAY  │  │ KEYBOARD │  │ CURSOR   │  │  HARD    │  │  WIRELES S/  │
│  821     │  │   822    │  │ CONTROL  │  │  COPY    │  │  TELEPHO NY  │
│          │  │          │  │ DEVICE   │  │ DEVICE   │  │ INTE RFACE   │
│          │  │          │  │   823    │  │   824    │  │    825       │
└──────────┘  └──────────┘  └──────────┘  └──────────┘  └──────────────┘
```

FIG. 8

FIG. 9

FIG. 10